Fine-Grained Access Control via Policy-Carrying Data

JULIAN A. PADGET, University of Bath, United Kingdom
WAMBERTO W. VASCONCELOS, University of Aberdeen, United Kingdom

We address the problem of associating access policies with datasets and how to monitor compliance via policy-carrying data. Our contributions are a formal model in first-order logic inspired by normative multi-agent systems to regulate data access, and a computational model for the validation of specific use cases and the verification of policies against criteria. Existing work on access policy identifies roles as a key enabler, with which we concur, but much of the rest focusses on authentication and authorization technology. Our proposal aims to address the normative principles put forward in Berners-Lee's bill of rights for the internet, through human-readable but machine-processable access control policies.

CCS Concepts: •**Security and privacy** → **Formal security models; Logic and verification;** *Information accountability and usage control;* •**Theory of computation** → **Modal and temporal logics;**

General Terms: Policies, Deontic Logic, Reasoning, Data Sharing

Additional Key Words and Phrases: Deontic logic, Data sharing

ACM Reference Format:
Julian A. Padget, Wamberto W. Vasconcelos, 2016. Fine-Grained Access Control via Policy-Carrying Data *ACM Trans. Internet Technol.* 0, 0, Article 00 (2016), 24 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Recent data-intensive research trends such as the Internet-of-Things (IoT) and Big Data, combined with socio-technical systems (STS) such as social networking and supported by portable devices (with sensors, GPS, etc.) make companies, research centres, and all of us, as individuals, both producers and consumers of data. A sensitive issue for data providers concerns control over access, sharing, dissemination and use of data. We regard control as placing restrictions on *who* can access the data, *when* data can be accessed, *how* data can be accessed, and so on. Although Berners-Lee does not provide a shopping-list of features in [Berners-Lee 1999, Ch.11], he sets out similar informal (and abstract) normative aims, stating:

> The Platform for Privacy Preferences Project (P3P) will give a computer a way of describing its owner's privacy preferences and demands, while it gives servers a way of describing their privacy policies, all implemented so that machines can understand each other and negotiate any differences.

P3P activity suspended in 2007, shortly after the publication of version 1.1 of the platform specification [P3P 2006], citing a lack of support from browser developers. The aim at the time appears to have been to support privacy in the context of consumer-to-business (purchasing) transactions via browser (consumer) and web-site (business).

Authors's addresses: J. A. Padget, Dept. of Computer Science, University of Bath, Bath, BA2 7AY, U.K., j.a.padget@bath.ac.uk. W. W. Vasconcelos (Corresponding author), Dept. of Computing Science, University of Aberdeen, Aberdeen, AB24 3LT, U.K. w.w.vasconcelos@abdn.ac.uk
W. W. Vasconcelos acknowledges the support of the Engineering and Physical Sciences Research Council (EPSRC, UK) within the research project "Scrutable Autonomous Systems" (SAsSY, http://www.scrutable-systems.org, Grant ref. EP/J012084/1).

DOI: 0000001.0000001

What may have seemed significant at the time, appears with hindsight to have a relatively narrow technological and use-case basis – to which [P3P 2006] is very specific – but equally, with hindsight, the same vision, issues and principles appear to be applicable in the emerging environments of IoT and STS.

Consider a scenario in which a health insurance company offers its customers a mobile phone app which collects data from a fitness wristband. The data collected concern blood pressure, heartbeat, amount of physical exercise and sleeping patterns; additionally it would also be possible to collect information on what people eat and drink via the app. The insurance company aims to offer better deals to customers who lead healthy lifestyles and, conversely, make more adequate provisions for customers with sedentary and disease-prone lifestyles. Users of the app, however, should have means to decide on the *policies* governing the data. For instance, even though users might agree to provide to the insurance company their heartbeat data (because they might get a reduced price when renewing their insurance) they may deny access to this data to any for-profit third-party (*e.g.*, a pharmaceutical company).

The current data landscape supports relative freedom of movement of data from individuals to the data silos used in cloud computing and thence between silos; this might contribute to the sense of lack of control which data providers might feel over their own data, privacy controls aside [Brandimarte et al. 2013]. This is further complicated as platforms may enable the collection and interpretation of those data, thus adding value to them. Our proposal associates data with bespoke policies: for example, framework policies might be defined by legislation, while specific policies for individual needs would have to satisfy the norms established at the primary level [Li et al. 2013].

In this paper we present an approach to represent fine-grained controls over data and to associate that inseparably from the data via what we call "policy-carrying data" (PCD[1]). Our PCDs explicitly represent who, when and how, also establishing what the consumer should (not) do when accessing data. Our proposal is novel in that we can establish permissions, obligations and prohibitions concerning what the consumer should (not) do when data are accessed; these permissions, obligations and prohibitions as well as the interconnections among different PCDs provide a foundation for transparency which is essential to a data-sharing economy. Obligations, prohibitions and permissions can be seen as transactional units in a non-pecuniary data economy, where access to and use of data may be traded for obligations, prohibitions and permissions that act as a form of user-definable, liquidity-at-point-of-use community currency [Litaer 2002]. These obligations, permissions and prohibitions may pertain directly to actions of data consumers or – and this is another significant novelty of our approach – indirectly to the policy associated with the extracted data or the data derived from them.

The main contributions of this paper are (i) a formal representation for PCD, with practicality concerns, and (ii) a reference implementation of core elements of our proposal. Additionally, we provide a computational context whereby stakeholders, processes and information model come together to share data via PCD. We build upon and extend the research presented in [Padget and Vasconcelos 2015]; however, whereas that paper was concerned with a much simpler propositional formalism, we have developed a more expressive first-order notation with practical concerns, that is, the mechanisms manipulating the formalisation are decidable and tractable. The implementation has not been previously reported.

[1]PCD also stands for "policy-carrying data collection" and we use PCDs (in the plural) to indicate a set of policy-carrying data collections.

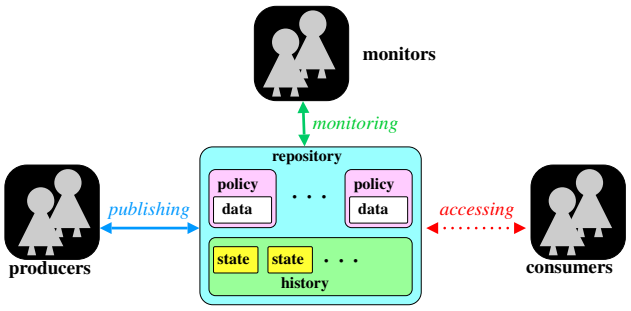

Fig. 1. Stakeholders, Processes & Information Model

We present in Section 2 how we envisage stakeholders and PCDs will come together in a computational setting. In Section 3 we present the syntax and (operational) semantics of our PCDs, and sketch some mechanisms using PCDs. In Section 4 we present a reference implementation of our approach. In Section 5 we discuss related work and we conclude the paper in Section 6.

2. POLICY-CARRYING DATA: STAKEHOLDERS, PROCESSES AND INFORMATION MODEL

We illustrate in Fig. 1 the stakeholders (squares with round edges), their processes (arrows), and an information model (boxes within central box) associated with our PCDs. The stakeholders are (i) data owners/producers who make data/information available (represented as the left-hand square); (ii) data consumers who want to access data (represented as the right-hand square); (iii) monitor/police who are responsible for monitoring/policing the publication and access activities (represented by the upper square in the middle). The first two types of stakeholders can be organisations or individuals as well as devices such as sensors, programs, databases, and so on. The monitor/police works as a third-party authority ensuring that activities (publishing and accessing) follow policies and dealing with violations.

Each of these stakeholders has their specific ways to interact via the repository: (i) **publishing** (represented by the blue solid arrow) is the process whereby data owners/producers make their data available but "wrapped" within a policy, that is, they publish, in a repository, some policy-carrying data (ii) **accessing** (represented by the red dotted arrow) is the process whereby data consumers *attempt* to obtain access to data mediated via policies (iii) **monitoring** (represented by the green arrow) concerns observing activities and checking for policy compliance or violation, and dispensing rewards or sanctions.

Our proposal relies on an information model (stored within the "repository" rectangle in the centre of the diagram) comprising the PCD (a policy and an associated data collection made available through the policy) and a history (a collection of events, *i.e.*, a record of activities carried out) gathered at particular time points, denoted as the *states* of the repository. This information model supports stakeholders carrying out the cycle of publish-access-monitor activities using a Web server equipped with functionalities to enable the policing of those accessing and uploading PCDs, keeping records of usage and (non-)compliance, and enforcing the policies' access control. We envisage programmatic access to PCDs, whereby programs and functionalities developed with specific technologies can access any PCD, interacting via pre-established protocols.

A typical PCD would express something like "Research staff can access 200 records of my data". If an interested party requested 500 records, the server would (i) check the

credentials of the requester (who needs to be registered); (ii) grant access to 200 records (a message would provide reasons for not providing the 500 records); (iii) update the record of that requester with respect to that PCD. Further requests from research staff would be rejected with a suitable justification. For such control to be in place, the server requires a record of events: an explicit account of the history of the PCD, how they have been used, by whom and when.

There are obvious similarities between our framework and existing approaches. Existing mechanisms to regulate resource access in distributed systems [Anderson 2001] have similar provisions as our framework – stakeholders, activities and (parts of) the information model – however as we show below, our policy language is more expressive, which in its turn, requires a more sophisticated information model allowing for extra functionalities to be in place. The language used to express policies clearly plays an important role in acceptability, accessibility and functionality. We put forward a model language, that is not tied to a concrete and standardised syntax, in Section 3. However, there are lessons to take from a wide variety of initiatives across the computer science domain as we discuss in Section 3 and more broadly in related work (Section 5).

3. A LANGUAGE FOR POLICIES-CARRYING DATA

There has been much research addressing data access policies, dating back from early UNIX file systems [Suhendra 2011; Tonti et al. 2003; Ferraiolo et al. 2011]. In our approach we include means to refer to a history of events, as in, for instance, "the first n users can access my data" and "anyone is permitted to use n records of my data". We provide fine-grained control over who is to access the data, and under what circumstances; for instance, "user u_2 is forbidden to access my data" and "anyone from company x may use my data after 6PM". We can also capture dynamic aspects of data usage, examples being "whoever accesses D_1 should not access D_2" and "anyone who uses my data should provide data". Although our formalism does not offer logical implication (to reduce the complexity of associated reasoning mechanisms), we provide means to relate data access/provision events via activation and deactivation conditions, which enables us to represent norms such as "anyone who uses my data should provide me with data".

We combine, adapt and extend existing proposals on normative (multi-agent) systems [Meneguzzi et al. 2015; Şensoy et al. 2012; García-Camino et al. 2009; Vasconcelos et al. 2009], representing data-related events (such as accessing records or publishing data collections), authorship of events and attempted actions, activation and deactivation conditions of policies, and the object of the policy, namely, the data collection itself. We introduce in the subsections below a language for policies and a representation for policy-carrying data, and equip these with a simple operational semantics using states and histories.

3.1. Underpinnings: a Fragment of First-Order Logic

Our building blocks are first-order predicates π of the form $p_i^n(t_1, \ldots, t_n)$ where p_i^n is a predicate symbol, n is the arity of the predicate symbol (omitted when the context makes it clear) and $t_j, 1 \leq j \leq n$, are variables (denoted as v, w, x, y, z, possibly with subscripts) or constants (denoted as a, b, c, d, possibly with subscripts). We make use of two logical operators, namely conjunction $\wedge$ and negation $\neg$, and define our formulae φ via the grammar $\varphi ::= \varphi \wedge \varphi \mid \pi \mid \neg\pi$. We note that in our language negation is only applicable to predicates π, and not to sub-formulae; moreover, negation cannot be nested. This means our language is less expressive than first-order logic and, in particular, we cannot define other operators such as disjunction $\vee$ and implication $\rightarrow$. This restriction in expressiveness enables us to provide computational mechanisms

which are decidable (unlike first-order logic) and of practical use, as explained below. We refer to all formulae of this fragment of first-order logic as $\mathcal{L}$.

Typical examples of first-order predicates are $access(d_1, u_1, temperature, 500)$, which intuitively states that the field "$temperature$" from data collection d_1 has been accessed by user u_1 500 times; and $provide(d_2, u_{455}, gps, 20)$, which states that user u_{455} provided 20 data items "gps" to data collection d_2.

Since we allow variables to appear in our formulae, we must consider their quantification. Let $vars(\pi) = \{x_0, \ldots, x_n\}$ be a function to obtain the possibly empty and finite set of variables $x_i, 0 \leq i \leq n$, in predicate π; we extend this function to obtain the variables of φ formulae: $vars(\varphi \wedge \varphi') = vars(\varphi) \cup vars(\varphi')$ and $vars(\neg\pi) = vars(\pi)$. We extend our φ formulae with the existential quantifier $\exists$ and the universal quantifier $\forall$, and we introduce a vector notation as a shorthand for convenience, $\vec{x} \stackrel{\text{def}}{=} x_0, \ldots, x_n$, to refer to all quantified variables in a particular order. Our quantified formulae are thus $\exists\vec{x}.\varphi$ and $\forall\vec{x}.\varphi$, where $vars(\varphi) = \{x_0, \ldots, x_n\}$. This means that all variables of a formula are in the scope of one same existential or universal quantifier, which prefixes a formula, that is, there is no nesting of quantifiers, and quantifiers must precede a formula (a quantifier cannot appear within sub-formulae).

In order to define our semantics, we make use of a unification operation "$\cdot$", associating a substitution $\sigma = \{x_0/t'_0, \ldots, x_m/t'_m\}$, that is, a possibly empty and finite set of pairs $x_i/t'_i, 0 \leq i \leq m$, as follows [Apt 1997; Fitting 1996]:

(1) $c \cdot \sigma = c$, that is, a constant c unified with any substitution is c itself
(2) $x \cdot \sigma = x$, iff $x/t'_i \notin \sigma$, that is, if x is not associated with any t'_i in σ, then its unification with σ is x itself.
(3) $x \cdot \sigma = t'_i \cdot \sigma$, iff $x/t'_i \in \sigma$, that is, the unification of x with a substitution in which x is associated with a term t'_i (that is, a variable or a constant) is the unification of t'_i with σ.
(4) $p(t_1, \ldots, t_n) \cdot \sigma = p(t_1 \cdot \sigma, \ldots, t_n \cdot \sigma)$, that is, the unification of a predicate with σ is the predicate with each of its terms unified with σ.
(5) $(\neg\pi) \cdot \sigma = \neg(\pi \cdot \sigma)$, that is, the unification of a negated predicate π with σ is the negation of the unification of π with σ.
(6) $(\varphi \wedge \varphi') \cdot \sigma = (\varphi \cdot \sigma \wedge \varphi' \cdot \sigma)$, that is, the unification of a conjunction $(\varphi \wedge \varphi')$ with σ is the conjunction of the unification $(\varphi \cdot \sigma \wedge \varphi' \cdot \sigma)$.

Substitutions can be *composed*, that is, given $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\sigma' = \{x'_1/t'_1, \ldots, x'_m/t'_m\}$ (where $\{x_1, \ldots, x_n\} \cap \{x'_1, \ldots, x'_m\} = \emptyset$), their composition, denoted as $\sigma \cdot \sigma'$, is the substitution $\{x_i/(t_i \cdot \sigma')\} \cup \sigma'$.

The semantics of our formulae is given in terms of a model (or interpretation) $\mathcal{S}$ comprising a possibly empty and finite set of ground atomic predicates, that is, predicates without variables – all their terms/parameters are constants. We shall denote a ground predicate as $\bar{\pi}$, and we note that for any predicate π and ground predicate $\bar{\pi}'$, we can obtain, in linear time, a substitution σ such that $\pi \cdot \sigma = \bar{\pi}'$ if the substitution exists; we can also find out, in linear time, if such substitution does not exist [Fitting 1996; Martelli and Montanari 1982].

We define below an *interpretation* relation $\mathbf{I}$, associating a model $\mathcal{S}$, a formula φ and a set of substitutions $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ as follows:

(1) $\mathbf{I}(\mathcal{S}, \pi, \{\sigma\})$ holds iff there is a $\bar{\pi}' \in \mathcal{S}$ such that $\pi \cdot \sigma = \bar{\pi}'$, that is, a predicate π holds in $\mathcal{S}$ under σ iff $\pi \cdot \sigma = \bar{\pi}'$ for some ground predicate $\bar{\pi}' \in \mathcal{S}$.
(2) $\mathbf{I}(\mathcal{S}, \neg\pi, \{\emptyset\})$ holds iff there is not one $\bar{\pi} \in \mathcal{S}$ such that $\pi \cdot \sigma = \bar{\pi}$, that is, the set of substitutions is just one empty substitution, as there is not one $\bar{\pi}' \in \mathcal{S}$ s.t. $\pi \cdot \sigma = \bar{\pi}$.
(3) $\mathbf{I}(\mathcal{S}, (\varphi \wedge \varphi'), \{\sigma\})$ holds iff $\mathbf{I}(\mathcal{S}, \varphi, \{\sigma\})$ and $\mathbf{I}(\mathcal{S}, \varphi', \{\sigma\})$ hold.
(4) $\mathbf{I}(\mathcal{S}, (\exists x_0, \ldots, x_n.\varphi), \{\sigma\})$ holds iff $\mathbf{I}(\mathcal{S}, \varphi, \{\sigma\})$ holds for at least one σ.

(5) $\mathbf{I}(\mathcal{S}, (\forall x_0, \ldots, x_n.\varphi), \{\sigma_1, \ldots, \sigma_m\})$ holds iff $\mathbf{I}(\mathcal{S}, \varphi, \{\sigma_i\}), 1 \leq i \leq m$, hold for every possible σ_i.

3.2. Policies as Atomic Deontic Formulae

We make use of first-order atomic deontic formulae [McNamara 2006; Meyer and Wieringa 1993; Meyer et al. 1994; von Wright 1951] in our PCD formulation; these are defined as follows:

Definition 3.1. A first-order atomic deontic formulae Δ is any construct of the form $\exists\vec{x}.\Box\pi$ and $\forall\vec{x}.\Box\pi$ where:
(1) $\vec{x} = x_0, \ldots, x_n$ is a (possibly empty and finite) vector (sequence) of variables.
(2) $\Box \in \{\mathsf{O}, \mathsf{F}, \mathsf{P}\}$ is one of the 3 deontic modalities O (for "obliged"), F (for "forbidden"), and P (for "permitted") representing, respectively, an obligation, a prohibition, and a permission.
(3) π is a first-order predicate such that $vars(\pi) = \{x_0, \ldots, x_n\}$

Typical examples of deontic atomic formulae are $\forall x.\mathsf{F}access(u_1, 1, x)$, establishing that user u_1 is forbidden to access one (any) record from any data collection x, and $\exists x.\mathsf{O}provide(u_{455}, 20, x)$, establishing that user u_{455} is obliged to provide 20 records to any one data collection x. Following the conventions of standard deontic logic [McNamara 2006; von Wright 1951], the modalities interrelate:

— $\mathsf{F}\pi \stackrel{\text{def}}{=} \mathsf{O}\neg\pi$, that is, a prohibition is an obligation on $\neg\pi$

— $\mathsf{P}\pi \stackrel{\text{def}}{=} \neg\mathsf{O}\neg\pi$, that is, a permission is the negation of an obligation on $\neg\pi$.

Although we only need one deontic modality (as the other two can be formally represented with it and the negation operator), in line with the body of work on deontic and normative research, we offer all three modalities, namely, permission P, prohibition F and obligation O, as it is easier to express and understand deontic formulae without nested negations. Quantification and modalities have been studied elsewhere (e.g., [Basin et al. 2010; Castellini 2005]). We show below, when we define an operational semantics, how quantifications and deontic modalities come together.

In our work we model existentially quantified obligations and permissions ($\exists\vec{x}.\mathsf{O}\pi$ and $\exists\vec{x}.\mathsf{P}\pi$, respectively) and universally quantified prohibitions ($\forall\vec{x}.\mathsf{F}\pi$). These deontic formulae capture common patterns of regulated behaviour [Meneguzzi et al. 2015], namely, an obligation is complied with if at least one instantiation of an action (with specific values) is carried out; permissions are also over specific values, especially when permissions are interpreted as exceptions to prohibitions (as in, for instance, [Şensoy et al. 2012]). Prohibitions, on the other hand, are normally established to rule out any instance of an action. We notice, however, that universally quantified deontic formulae may containt constants and hence we can also represent prohibitions on specific actions. Finally, it is worth mentioning that there is no technical reason not to allow using the deontic operators with any quantification, but for simplicity, ease of presentation, and pragmatic reasons, we only consider some combinations.

We introduce our policies via Def. 3.2; these are the "policy" part of our PCDs:

Definition 3.2. A policy Π is of the form $\langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \Delta\rangle$ where:
(1) $\varphi^a, \varphi^d \in \mathcal{L}$ are formulae of our first-order fragment and which represent activation and deactivation conditions, respectively;
(2) Δ is an atomic deontic logic formula (cf. Def. 3.1).

We do not allow nesting of quantifiers so as to simplify the language which underpins our approach. However, we use our policies as *rules* [Buchanan and Duda 1983; García-Camino et al. 2009; Meneguzzi et al. 2015], this becoming obvious in our operational semantics below. We allow variables appearing in φ^a to also appear in φ^d and Δ, and

the formalisation above is in fact a shorthand for $\forall\vec{x}.((\varphi^a \wedge \neg(\exists\vec{y}.\varphi^d)) \rightarrow \Delta)$, where $\rightarrow$ is the standard material implication, that is, $\varphi \rightarrow \varphi'$ if, and only if, $\neg\varphi \vee \varphi'$. Such formulation has been adopted by various approaches to normative multi-agent systems (e.g., [García-Camino et al. 2009; Şensoy et al. 2012; Meneguzzi et al. 2015]).

The semantics of policies builds on the interpretation relation for our first-order fragment: $\mathbf{I}(\mathcal{S}, \langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \Delta\rangle, \{\sigma_1, \ldots, \sigma_m\})$ holds iff:

(1) $\mathbf{I}(\mathcal{S}, \varphi^a, \{\sigma_i\}), 1 \leq i \leq m$, holds for every possible σ_i, and
(2) $\mathbf{I}(\mathcal{S}, \varphi^d \cdot \sigma_i, \{\sigma\}), 1 \leq i \leq m$, does not hold for any σ.

Case 1 above establishes all *instances* of the activation condition/formula φ^a which arise from the model $\mathcal{S}$. Case 2 states that we must check that none of the various instances of deactivation conditions $\varphi^d \cdot \sigma_i$ (one for each unification σ_i of the activation condition in the model) holds, that is, we cannot find σ in $\mathcal{S}$ such that $\mathbf{I}(\mathcal{S}, \varphi^d \cdot \sigma_i, \{\sigma\})$ holds. Additionally, the semantics above captures the instances of the deontic formulae: let Δ be of the form $\exists z.\Box\pi$ (cf. Def. 3.1), then the semantics above provides the set of instances $\{\exists z.\Box(\pi \cdot \sigma_i) | \mathbf{I}(\mathcal{S}, \varphi^a, \{\sigma_i\}), 1 \leq i \leq m\}$; a similar set of instances is defined for Δ of the form $\forall z.\Box\pi$.

3.3. Policy-Carrying Data

PCDs are formally defined as Def. 3.3: we are not specific about what the data collections are – these can be individual records of a database, files, readings from a sensor, and so on. Very importantly, rather than having data collections replicated in every PCD referring to them, there could be only one copy of the data collection and all PCDs regulating its access would make use of a unique locator such as a URL.

Definition 3.3. A policy-carrying data (collection) PCD is a pair $\langle\Pi, D\rangle$ where Π is a policy (cf. Def. 3.2) and $D = \{d_1, \ldots, d_n\}$ is a set of data items.

We use data collection and data interchangeably; PCD stands both for "policy-carrying data" and "policy-carrying data collection", although the latter can be used in the plural (PCDs standing for "policy-carrying data collections").

We make use of a subset of first-order predicates to create a vocabulary of action labels $\mathbf{Act}$ which are the target of the policies. An action predicate $\pi^{\mathbf{Act}}$ is one of the following (with their intuitive meaning)[2]:

— $access(x, y, z)$ – x has accessed y records of data collection z.
— $provide(x, y, z)$ – x has provided y records of data collection z.

We adapt Defs. 3.1–3.3 to reflect this: our deontic formulae are represented as $\Delta^{\mathbf{Act}}$ and are of the form $\exists\vec{x}.\Box\pi^{\mathbf{Act}}$ or $\forall\vec{x}.\Box\pi^{\mathbf{Act}}$; our policies, represented as $\Pi^{\mathbf{Act}}$, are of the form $\langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \Delta^{\mathbf{Act}}\rangle$, and a PCD is of the form $\langle\Pi^{\mathbf{Act}}, D\rangle$. When no confusion arises we shall omit the $\mathbf{Act}$ superscript for simplicity.

A sample policy using action labels is

$$\langle\forall x.\neg access(x, 50, temp), access(x, 50, temp), \mathsf{P}access(x, 50, temp)\rangle$$

This establishes that anyone (referred to by the universally quantified variable x) is permitted to access 50 records of data collection $temp$; the norm is activated if the records haven't yet been accessed, and the norm is deactivated when 50 records are accessed. We explain below that policies are instantiated to individuals: although the policy is stated in general terms, for policing/monitoring purposes (and for sanctioning/rewarding when this is the case), we must keep a record of individuals' activities and the policies which are applicable to them (via their roles). We explain below how roles are captured. The deactivation condition and deontic formula above are shown

[2]We note that the action predicates can be more sophisticated, including, for instance, a description of the kinds of records and fields of a data collection someone can access or provide.

without quantifiers as their only variable x appears universally quantified in the activation condition.

Roles enable the generic reference to individuals with similar social or organisational status, standing or credentials [Biddle 1979; Turner 2001]; role-based access control models [Sandhu et al. 1996; Suhendra 2011] refer to groups of users via their roles. Some approaches [Padget and Vasconcelos 2015; Vasconcelos et al. 2009, 2012] annotate the deontic modality with the role r which the policy is aimed at, as in, for instance, $\mathsf{O}_r\pi$. However, the same effect can be achieved by adding a predicate $role(x, r)$ (establishing that individual x has role r) in the activation condition of a policy, that is, $\langle \forall \vec{x}.\varphi^a, \exists \vec{y}.\varphi^d, \exists \vec{z}.\Box_r\pi \rangle$ is a shorthand for $\langle \forall \vec{x}.(\varphi \wedge role(x, r)), \exists \vec{y}.\varphi^d, \exists \vec{z}.\Box\pi \rangle$ (and similarly for $\forall \vec{z}.\Box_r\pi\rangle$). We use a finite and non-empty set of role labels $R = \{r_1, \ldots, r_t\}$ and we assume a finite and non-empty set of individuals $A = \{a_1, \ldots, a_s\}$ uniquely identified. Some roles can be associated with individuals through their membership to organisations (*i.e.*, institutions or companies). We assume that individuals have their credentials appropriately recorded (in our states or "snapshots" as explained below) by those providing the data sharing setup, and these credentials are used when checking the applicability of policies.

3.4. Operational Semantics

In this section we explain the operational semantics connecting the syntax and semantics of our policies with an underlying computational model. Our underlying computational model is a sequence of *states*. A state is represented as the model $\mathcal{S}$ introduced in our interpretation relation above, and provides a "snapshot" of actual events; each event is recorded as a ground predicate $\bar{\pi}$. Similar models have been previously proposed (e.g., [García-Camino et al. 2009; Fisher 2006]) and, as we show below, are closely related to the formal semantics of modal logics. For compactness (and to avoid having to check for consistency), we do not record negated predicates in our states, thus adopting the *closed world assumption* [Reiter 1978] which establishes that what is not stated/proven as true is deemed false.

A sequence of states represents a *history*: histories record sequences of states, providing a linear account of events and how they are temporally related. A history $\mathcal{H} = \langle \mathcal{S}_0, \ldots, \mathcal{S}_n \rangle$ is a possibly empty and finite sequence of states $\mathcal{S}_j, 0 \leq j \leq n$. We formally connect policies with histories. We define below means to check if a policy was active in a history.

Definition 3.4. A policy $\Pi^{\mathbf{Act}} = \langle \forall \vec{x}.\varphi^a, \exists \vec{y}.\varphi^d, \Delta^{\mathbf{Act}} \rangle$ was active in history $\mathcal{H} = \langle \mathcal{S}_1, \ldots, \mathcal{S}_n \rangle$ under substitutions σ^a and σ^d if, and only if, the following conditions hold:

(1) $\mathbf{I}(\mathcal{S}_1, \forall \vec{x}.\varphi^a, \{\sigma^a\})$ holds for some σ^a, that is, the policy became active (the activation condition holds) at state 1,

(2) $\mathbf{I}(\mathcal{S}_n, \exists \vec{y}.\varphi^d \cdot \sigma^a, \{\sigma^d\})$ holds for some σ^d, that is, the policy became inactive (the deactivation condition holds) at state n, and

(3) $\mathbf{I}(\mathcal{S}_i, \exists \vec{y}.\varphi^d \cdot \sigma^a, \Sigma), 1 < i < n$, does not hold, that is, the policy was not deactivated in the intervening states.

We represent policy activation as the relation $active(\Pi^{\mathbf{Act}}, \mathcal{H}, \sigma^a, \sigma^d)$. We note that there might be many σ^a for one same policy and state, representing the "customisation" of a policy to a specific context.

We establish the conditions for policy *compliance* with the three definitions below.

Definition 3.5. A policy $\Pi^{\mathbf{Act}} = \langle \forall \vec{x}.\varphi^a, \exists \vec{y}.\varphi^d, \exists \vec{z}.\mathsf{O}\pi^{\mathbf{Act}} \rangle$ (an existential obligation) was complied with in history $\mathcal{H} = \langle \mathcal{S}_1, \ldots, \mathcal{S}_n \rangle$, under substitutions σ^a, σ^d, denoted as $comply^{\mathsf{O}}(\Pi^{\mathbf{Act}}, \mathcal{H}, \sigma^a, \sigma^d)$, if, and only if, the following conditions hold:

(1) $active(\Pi^{\mathbf{Act}}, \mathcal{H}, \sigma^a, \sigma^d)$, that is, the policy was active in the history under σ^a and σ^d.

(2) $\mathbf{I}(\mathcal{S}_j, \exists\vec{z}.(\pi^{\mathbf{Act}} \cdot \sigma^a), \{\sigma^\Delta\})$ holds for some state $\mathcal{S}_j, 1 < j \leq n$, and σ^Δ, that is, there is a $\bar{\pi}^{\mathbf{Act}} \in \mathcal{S}_j$ such that $\bar{\pi}^{\mathbf{Act}} = (\pi^{\mathbf{Act}} \cdot \sigma^a) \cdot \sigma^\Delta$.

As an example $\langle(\forall x.access(x, 20, D_1)), (\exists yz.provide(x, y, z)), (\exists yz.\mathsf{O}provide(x, y, z))\rangle$ has activation condition "anyone accessing 20 records of D_1"; when the policy is active the same people who accessed the records are obliged to provide records to some data collection. The policy is deactivated when some records are provided. A history in which this policy is complied with is:

$$\mathcal{H} = \langle\overbrace{\{access(bob, 20, D_1)\}}^{\mathcal{S}_1}, \overbrace{\{provide(bob, 10, D_2)\}}^{\mathcal{S}_2}\rangle$$

The policy was active in the history (cf. Def. 3.4) as $\mathcal{S}_1$ fulfills the activation condition, $\mathcal{S}_2$ fulfills the deactivation condition and there are no intermediary states. Moreover, the activation condition instantiates via $\sigma^a = \{x/bob\}$ the obligation $\exists yz.\mathsf{O}provide(bob, y, z)$. $\mathcal{S}_2$ is also state $\mathcal{S}_j$ of case 2 in Def. 3.5 where the obligation is fulfilled, as we have $\bar{\pi} = provide(bob, 10, D_2)$ and $\sigma^\Delta = \{y/10, z/D_2\}$.

Definition 3.6. A policy $\Pi^{\mathbf{Act}} = \langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \forall\vec{z}.\mathsf{F}\pi^{\mathbf{Act}}\rangle$ (a universal prohibition) was complied with in history $\mathcal{H} = \langle\mathcal{S}_1, \ldots, \mathcal{S}_n\rangle$, under substitutions σ^a and σ^d, denoted as $comply^{\mathsf{F}}(\Pi^{\mathbf{Act}}, \mathcal{H}, \sigma^a, \sigma^d)$, if, and only if, the following conditions hold:
(1) $active(\Pi^{\mathbf{Act}}, \mathcal{H}, \sigma^a, \sigma^d)$, that is, the policy was active in the history under σ^a and σ^d.
(2) $\mathbf{I}(\mathcal{S}_j, \exists\vec{z}.(\pi^{\mathbf{Act}} \cdot \sigma^a), \Sigma)$ does not hold for any state $\mathcal{S}_j, 1 < j \leq n$, that is, there is not one $\bar{\pi}^{\mathbf{Act}} \in \mathcal{S}_j, 1 < j \leq n$, such that $\bar{\pi}^{\mathbf{Act}} = (\pi^{\mathbf{Act}} \cdot \sigma^a) \cdot \sigma$ for any σ.

Policy $\langle(\forall xyz.\neg provide(x, y, z)), (\exists x'y'z'.provide(x', y', z')), (\mathsf{F}access(x, y, D_1))\rangle$, for example, establishes that anyone who has not provided any records to any data collection is forbidden to access records from D_1. The policy is deactivated when someone provides some records. A history in which this policy is complied with is:

$$\mathcal{H} = \langle\overbrace{\emptyset}^{\mathcal{S}_1}, \overbrace{\{provide(bob, 10, D_2)\}}^{\mathcal{S}_2}\rangle$$

The policy was active in the history (cf. Def. 3.4) as $\mathcal{S}_1$ fulfills the activation condition, $\mathcal{S}_2$ fulfills the deactivation condition and there are no intermediary states. Since there are no states $\mathcal{S}_j$ in which $(access(x, y, D_1) \cdot \sigma^a) \cdot \sigma$ occurs, the policy was complied with.

In data sharing scenarios, permissions are very important as they establish explicit access rights, asserting that what is not explicitly permitted (that is, there is not an active permission addressing a particular action) is forbidden. Moreover, permissions can be seen as *exceptions* to prohibitions and obligations, along the lines of, e.g., [Boella and van der Torre 2003; Governatori et al. 2013]. We provide a means to check the compliance of a set of permissions:

Definition 3.7. A set of policies $\mathbf{\Pi}^{\mathbf{Act}} = \{\Pi_1^{\mathbf{Act}}, \ldots, \Pi_m^{\mathbf{Act}}\}, \Pi_i^{\mathbf{Act}} = \langle\forall\vec{x}_i.\varphi_i^a, \exists\vec{y}_i.\varphi_i^d, \exists\vec{z}_i.\mathsf{P}\pi_i^{\mathbf{Act}}\rangle, 1 \leq i \leq m$ (all existential permissions), was complied with in history $\mathcal{H} = \langle\mathcal{S}_1, \ldots, \mathcal{S}_n\rangle$, under a set $\mathbf{\Sigma}$ of pairs of substitutions $\langle\sigma^a, \sigma^d\rangle$, denoted as $comply^{\mathsf{P}}(\mathbf{\Pi}^{\mathbf{Act}}, \mathcal{H}, \mathbf{\Sigma})$, if, and only if, for every $\bar{\pi}^{\mathbf{Act}} \in \mathcal{S}_j, 1 < j \leq n$, the following conditions hold:
(1) there is a $\Pi_k^{\mathbf{Act}} \in \mathbf{\Pi}^{\mathbf{Act}}, \Pi_k^{\mathbf{Act}} = \langle\forall\vec{x}_k.\varphi_k^a, \exists\vec{y}_k.\varphi_k^d, \exists\vec{z}_k.\mathsf{P}\pi_k^{\mathbf{Act}}\rangle, active(\Pi_k^{\mathbf{Act}}, \mathcal{H}', \sigma_k^a, \sigma_k^d)$, that is, a policy $\Pi_k^{\mathbf{Act}}$ was active in a sub-history $\mathcal{H}'$ of $\mathcal{H}$, under σ_k^a and σ_k^d. $\mathcal{H} = \mathcal{H}_1 \bullet \mathcal{H}' \bullet \mathcal{H}_2$, where "$\bullet$" is the concatenation operator for sequences of states, and $\mathcal{H}_1, \mathcal{H}_2$ are possibly empty sub-histories. Moreover, $\mathcal{H}' = \mathcal{H}_1' \bullet \langle\mathcal{S}_j\rangle \bullet \mathcal{H}_2'$, (where $\mathcal{H}_1', \mathcal{H}_2'$ are possibly empty sub-histories), that is, $\Pi_k^{\mathbf{Act}}$ was active in $\mathcal{S}_j$.
(2) $\bar{\pi}^{\mathbf{Act}} = (\pi_k^{\mathbf{Act}} \cdot \sigma_k^a) \cdot \sigma_j$ for some σ_j, that is, $\bar{\pi}^{\mathbf{Act}}$ is the target of $\Pi_k^{\mathbf{Act}}$ (activated with σ_k^a) and (possibly) further instantiated via σ_j.
If, and only if, σ_k^a, and σ_k^d are as above, $\langle\sigma_k^a, \sigma_k^d\rangle \in \Sigma$.

Def. 3.7 establishes that all $\bar{\pi}^{\mathbf{Act}} \in \mathcal{S}_j$ (all actions recorded in any state $\mathcal{S}_j$ of history $\mathcal{H}$) must be unifiable with $\pi_k^{\mathbf{Act}} \cdot \sigma_k^a$ of a permission $\Pi_k^{\mathbf{Act}}$ which was active at $\mathcal{S}_j$. There might be more than one such permission active, and there might be more than one σ_j for one permission and state. We illustrate Def. 3.7 with permissions $\mathbf{\Pi}^{\mathbf{Act}} = \{\Pi_1^{\mathbf{Act}}, \Pi_2^{\mathbf{Act}}, \Pi_3^{\mathbf{Act}}\}$:

$$\Pi_1^{\mathbf{Act}} = \langle \forall xz.user(x) \wedge data(z), endOfDay, \exists y.\mathsf{P}provide(x,y,z) \rangle$$
$$\Pi_2^{\mathbf{Act}} = \langle \forall xy.provide(x,y,D_1), endOfDay, \mathsf{P}access(x,y,D_2) \rangle$$
$$\Pi_3^{\mathbf{Act}} = \langle \forall xy.provide(x,y,D_1), endOfDay, \mathsf{P}access(x,y,D_3) \rangle$$

Where $endOfDay$ is a "flag", recorded by the administrators of the data sharing framework to indicate the end of a period of time. $\Pi_1^{\mathbf{Act}}$ establishes that any user x is permitted to provide any number of records y to any data collection z. $\Pi_2^{\mathbf{Act}}$ establishes that any x who provides y records to data collection D_1 is permitted to access the same number of records from D_2. $\Pi_3^{\mathbf{Act}}$ is similar, but the permission is for accessing data from D_3. A history $\mathcal{H}$ in which these policies are complied with is:

$$\langle \overbrace{\left\{ \begin{array}{c} user(bob), user(john), \\ data(D_1), data(D_2), \\ data(D_3) \end{array} \right\}}^{\mathcal{S}_1}, \overbrace{\left\{ \begin{array}{l} provide(bob, 10, D_1), \\ provide(john, 10, D_1) \end{array} \right\}}^{\mathcal{S}_2}, \overbrace{\{\ access(bob, 10, D_2)\ \}}^{\mathcal{S}_3}, \overbrace{\{endOfDay\}}^{\mathcal{S}_4} \rangle$$

We have $active(\Pi_1^{\mathbf{Act}}, \mathcal{H}, \{x/bob, z/D_1\}, \emptyset)$, $active(\Pi_1^{\mathbf{Act}}, \mathcal{H}, \{x/john, z/D_1\}, \emptyset)$, as well as other cases when z unifies with D_2 and D_3. We also have $active(\Pi_2^{\mathbf{Act}}, \langle \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \rangle, \{x/bob, y/10\}, \emptyset)$, and $active(\Pi_3^{\mathbf{Act}}, \langle \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \rangle, \{x/john, y/10\}, \emptyset)$. The ground predicate $\bar{\pi}^{\mathbf{Act}} \in \mathcal{S}_3$ is the target of $\Pi_2^{\mathbf{Act}}$ which is active, so the set of policies is complied with. An interesting situation arises in this scenario[3]: if the compliance check had been defined for one policy (instead of a set of policies), then $\Pi_3^{\mathbf{Act}}$, active in $\mathcal{S}_3$ and establishing $\mathsf{P}access(john, 10, D_2)$, would not unify with $\bar{\pi}^{\mathbf{Act}} = access(bob, 10, D_2)$ and a violation would occur. We avoid such situations with our definition as it establishes the compliance of permissions as a test to ensure any action performed is the target of an active permission. We note that we detect the violation of a *set* of permissions: whereas an obligation or a prohibition can be checked for compliance in isolation, checking the compliance of permissions requires all permissions to be considered together.

A generic definition of compliance, $comply(\mathbf{\Pi}, \mathcal{H}, \Sigma), \mathbf{\Pi} = \mathbf{\Pi}^{\mathsf{O}} \cup \mathbf{\Pi}^{\mathsf{F}} \cup \mathbf{\Pi}^{\mathsf{P}}$ (obligations $\mathbf{\Pi}^{\mathsf{O}}$, prohibitions $\mathbf{\Pi}^{\mathsf{F}}$ and permissions $\mathbf{\Pi}^{\mathsf{P}}$), holds if, and only if, the following hold: (1) $comply^{\mathsf{O}}(\mathbf{\Pi}^{\mathsf{O}}, \mathcal{H}, \Sigma^{\mathsf{O}})$, (2) $comply^{\mathsf{F}}(\mathbf{\Pi}^{\mathsf{F}}, \mathcal{H}, \Sigma^{\mathsf{F}})$, and (3) $comply^{\mathsf{P}}(\mathbf{\Pi}^{\mathsf{P}}, \mathcal{H}, \Sigma^{\mathsf{P}})$; moreover, $\Sigma = \Sigma^{\mathsf{O}} \cup \Sigma^{\mathsf{F}} \cup \Sigma^{\mathsf{P}}$. We extend Def. 3.5 for sets: $comply^{\mathsf{O}}(\mathbf{\Pi}^{\mathsf{O}}, \mathcal{H}, \Sigma^{\mathsf{O}}), \mathbf{\Pi}^{\mathsf{O}} = \{\Pi_1^{\mathsf{O}}, \ldots, \Pi_n^{\mathsf{O}}\}$, holds if, and only if, $comply^{\mathsf{O}}(\Pi_i^{\mathsf{O}}, \mathcal{H}'_{[i,j]}, \sigma^a_{[i,j]}, \sigma^d_{[i,j]})$ for all $i, 1 \leq i \leq n$, and all sub-histories $\mathcal{H}'_{[i,j]}, 1 \leq j \leq m_i$, of $\mathcal{H}$ in which Π_i^{O} was active, $active(\Pi_i^{\mathsf{O}}, \mathcal{H}'_{[i,j]}, \sigma^a_{[i,j]}, \sigma^d_{[i,j]})$, $\Sigma^{\mathsf{O}} = \cup_{i=1}^{n} \cup_{j=1}^{m_i} \{\langle \sigma^a_{[i,j]}, \sigma^d_{[i,j]} \rangle\}$. Def. 3.6 is extended in a similar fashion. A set of policies has been violated, $violated(\mathbf{\Pi}, \mathcal{H}, \Sigma)$ if, and only if, $comply(\mathbf{\Pi}, \mathcal{H}, \Sigma)$ does not hold, that is, for at least one $\Pi \in \mathbf{\Pi}$ the first condition (respectively, for obligations, prohibitions and permissions) of Defs. 3.5–3.7 holds and the second condition does not hold[4].

In open systems autonomous software agents are free to actually perform forbidden actions, but in a data-sharing context we want to rule out any policy-violating

[3]We thank an anonymous reviewer for pointing this out to us.

[4]We note that prohibitions and permissions can be checked for violation without a history – it is sufficient to check that the policy was active when the violation occurred (that is, a forbidden or a non-permitted action was carried out when the policy was active). To check the violation of an obligation, however, requires the history during which the policy was active and expired as only then we can establish that the obliged action was not carried out within the period of activation.

behaviour. We thus consider an *attempt* to access data as evidence of policy violation: consumers may try to access data they are not entitled to, and this attempt counts as if the data had been accessed, even though our PCD will prevent this from happening. Our policy violation above is interpreted under this light: the prohibited event is recorded but it did not actually happen.

3.5. Deontic Logic and Operational Semantics

The operational semantics provides a counterpart to the usual Kripke semantics used in (modal) deontic logics [McNamara 2006]. This enables us to draw parallels between deontic equivalences and relationships among our policies. We show that our operational semantics preserves an important result of our quantified deontic logic:

Claim 1. If $\exists\vec{x}.\mathsf{O}\pi$ does not hold then $\forall\vec{x}.\mathsf{F}\pi$ holds:

Proof:

1. $\neg(\exists\vec{x}.\mathsf{O}\pi)$ holds, then (premise $\exists\vec{x}.\mathsf{O}\pi$ does not hold, hence its negation holds)
2. $\neg(\exists\vec{x}.\neg\mathsf{O}\neg\pi)$ holds, then (axiom 3 of Standard Deontic Logic [McNamara 2006])
3. $\neg(\neg\forall\vec{x}.\mathsf{O}\neg\pi)$ holds, then (negation over quantification)
4. $\forall\vec{x}.\mathsf{O}\neg\pi$ holds, then (cancellation of double negation)
5. $\forall\vec{x}.\mathsf{F}\pi$ holds (by definition)

■

We prove below that this result also holds in our operational model. The $violated$ relation in our operational model corresponds to "not holding". Without loss of generality, we assume that our policies have the same activation condition and deactivation conditions and thus are active or not in exactly the same histories. This means condition 1 ($active(\Pi, \mathcal{H}, \sigma^a, \sigma^d)$) of Defs. 3.5–3.7 holds, and so does $active(\Pi, \mathcal{H}, \sigma^a, \sigma^d)$ in the definition of violation; thus we only need to check if compliance happened (or not).

Claim 2. If an existential obligation $\Pi^{\mathsf{O}} = \langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \exists\vec{z}.\mathsf{O}\pi\rangle$ was violated (does not hold) in history $\mathcal{H} = \langle\mathcal{S}_1, \ldots, \mathcal{S}_n\rangle$, $violated(\Pi^{\mathsf{O}}, \mathcal{H}, \sigma^a, \sigma^d)$, then the universal prohibition $\Pi^{\mathsf{F}} = \langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \forall\vec{z}.\mathsf{F}\pi\rangle$ was complied with (holds), $comply(\Pi^{\mathsf{F}}, \mathcal{H}, \sigma^a, \sigma^d)$.

Proof: If Π^{O} has been violated then $comply^{\mathsf{O}}(\Pi^{\mathsf{O}}, \mathcal{H}, \sigma^a, \sigma^d)$ does not hold (case 2, Def. 3.5), that is, $\mathbf{I}(\mathcal{S}_j, \exists\vec{z}.(\pi \cdot \sigma^a), \{\sigma^{\Delta}\})$ does not hold for any state $\mathcal{S}_j, 1 < j \leq n$, this means that there is not one $\bar{\pi} \in \mathcal{S}_j, 1 < j \leq n$, such that $\bar{\pi} = (\pi \cdot \sigma^a) \cdot \sigma^{\Delta}$ for any σ^{Δ}. This is precisely condition 2 of Def. 3.6 describing when $\Pi^{\mathsf{F}} = \langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \forall\vec{z}.\mathsf{F}\pi\rangle$ is complied with. ■

3.6. PCDs and Individual Agents

PCDs are ultimately aimed at individuals, although they are specified in general terms. The credentials (roles) referred to in a policy are ultimately of individual agents; actions are performed by agents, this being captured by the first argument of predicate $\pi^{\mathbf{Act}}$. Since we only consider states with fully ground atomic predicates, we can define a function to provide the agent a responsible for performing $\bar{\pi}^{\mathbf{Act}} \in \mathcal{S}$:

(1) $perf(access(a, n, d), \mathcal{S}) = a$, if $access(a, n, d) \in \mathcal{S}$

(2) $perf(provide(a, n, d), \mathcal{S}) = a$, if $provide(a, n, d) \in \mathcal{S}$

The compliance definitions (Defs. 3.5–3.7) can be extended to obtain the identity of individual agents responsible for complying with the policy. Given $\Pi = \langle\forall\vec{x}.\varphi^a, \exists\vec{y}.\varphi^d, \exists\vec{z}.\mathsf{O}\pi\rangle$ (an existential obligation) and a history $\mathcal{H} = \langle\mathcal{S}_1, \ldots, \mathcal{S}_n\rangle$ such that $comply^{\mathsf{O}}(\Pi, \mathcal{H}, \sigma^a, \sigma^d)$; we have $\bar{\pi} \in \mathcal{S}_j$ such that $\bar{\pi} = (\pi \cdot \sigma^a) \cdot \sigma^{\Delta}$, $perf(\bar{\pi}, \mathcal{S}_j) = a$, and similarly for permissions. For prohibitions, however, the agents who complied are all those which did not perform a prohibited action. We denote the compliance of an individual a to a set of policies $\mathbf{\Pi}$ in history $\mathcal{H}$ as $comply(\mathbf{\Pi}, \mathcal{H}, a)$. Since more than one agent may comply with the policies, we can compute them all as $complyAll(\mathbf{\Pi}, \mathcal{H}, A')$, $A' \subseteq A$, such that, for all $a \in A'$, $comply(\mathbf{\Pi}, \mathcal{H}, a)$.

$$\langle \overbrace{\langle \underbrace{\forall xy.\neg access(x,y,D_1)}_{activation}, \underbrace{\exists x'y'.access(x',y',D_1)}_{deactivation}, \underbrace{\exists x''y''.\mathsf{P}access(x'',y'',D_1)}_{target} \rangle}^{policy}, \overbrace{D_1}^{data} \rangle \quad (1)$$

$$\langle \langle \forall xy.access(x,y,D_1), endOfDay, \mathsf{F}access(x,y,D_2) \rangle, D_2 \rangle \quad (2)$$

$$\langle \langle \forall xy.access(x,y,D_1), provide(x,300,D_2), \mathsf{O}provide(x,300,D_2) \rangle, D_2 \rangle \quad (3)$$

Fig. 2. Sample PCDs

In realistic settings we need to consider longer histories in which a policy is complied with or violated many times. Using the operator "$\bullet$" to merge/split histories, we say that $\mathcal{H} = \mathcal{H}_1 \bullet \mathcal{H}_2 \bullet \cdots \bullet \mathcal{H}_n$ holds iff $\mathcal{H}_i = \langle \mathcal{S}^i_1, \ldots, \mathcal{S}^i_{m_i} \rangle, 1 \leq i \leq n$, and $\mathcal{H} = \langle \mathcal{S}^1_1, \ldots, \mathcal{S}^1_{m_1}, \mathcal{S}^2_1, \ldots, \mathcal{S}^2_{m_2}, \ldots, \mathcal{S}^n_1, \ldots, \mathcal{S}^1_{m_n} \rangle$. With this operator, we can compute, given a history, all the sub-histories in which a set of policies was complied with (or violated): $comply^*(\mathbf{\Pi}, \mathcal{H}, \{\mathcal{H}_1, \ldots, \mathcal{H}_p\})$ holds iff $\mathcal{H} = \mathcal{H}' \bullet \mathcal{H}_i \bullet \mathcal{H}''$, $comply(\mathbf{\Pi}, \mathcal{H}_i, \mathbf{\Sigma})$.

A similar computation can be defined for violations: $violated^*(\mathbf{\Pi}, \mathcal{H}, \{\mathcal{H}_1, \ldots, \mathcal{H}_p\})$ holds if, and only if, $\mathcal{H} = \mathcal{H}' \bullet \mathcal{H}_i \bullet \mathcal{H}''$, $violated(\mathbf{\Pi}, \mathcal{H}_i, \mathbf{\Sigma})$. We also define means to compute those individuals responsible for policy compliance/violation: $comply^*(\mathbf{\Pi}, \mathcal{H}, \{\mathcal{H}_1, \ldots, \mathcal{H}_p\}, \{a_{\mathcal{H}_1}, \ldots, a_{\mathcal{H}_p}\})$ if, and only if, for all $i, 1 \leq i \leq p$, $comply(\mathbf{\Pi}, \mathcal{H}_i, a_{\mathcal{H}_i})$. Again, there might be more than one agent responsible for policy compliance/violation in each sub-history, and we can obtain these as $complyAll^*(\mathbf{\Pi}, \mathcal{H}, \{\mathcal{H}_1, \ldots, \mathcal{H}_p\}, \{A_{\mathcal{H}_1}, \ldots, A_{\mathcal{H}_p}\})$ where for all $i, 1 \leq i \leq p, A_{\mathcal{H}_i} \subseteq A$, $complyAll^*(\mathbf{\Pi}, \mathcal{H}_i, A_{\mathcal{H}_i})$. With these basic operations, we can define policing mechanisms to dispense rewards and sanctions to individuals, based on histories of states and policies; we discuss one such mechanism below.

We make use of our formalism to represent typical examples of PCD; these are shown in Fig. 2. PCD (1) captures a simple permission for anyone to access all records of a data collection. The activation condition establishes that the permission is in place if no records have yet been accessed, and the policy is deactivated if anyone accesses any records, that is, the policy stipulates a "one-off" access to the data. PCD (2) illustrates a useful way to inter-relate policies. It states that anyone who accesses D_1 is forbidden to access D_2. In the PCDs in Fig. 2 we omitted quantifiers whose variables are already quantified, i.e., x, y in the deontic formula of PCD (2), and x in the deactivation condition and in the deontic formula of PCD (3). This creates a "chain" of events relating PCDs: if someone makes use of the permission to access D_1 (established by PCD (1)) then it is forbidden to access D_2. We also specify PCD (3), stating that those who access D_1 are obliged to provide 300 records to (be added to) D_2. The obligation is deactivated after the agent who accessed D_1 provides some data.

3.7. Reasoning with/about PCDs

In [Padget and Vasconcelos 2015] we present three mechanisms to enable stakeholders to reason with and about their PCDs. Although those mechanisms were aimed at a simpler (propositional) language, we claim that they can be easily extended to accommodate our first-order logic. Our argument to support this claim lies in the fact that our states are sets of fully ground predicates, and that detection of policy compliance/violation amounts to checking if predicates occur (or not occur) in states.

In that paper, we describe a process whereby publishers of PCDs can obtain the identity of individual agents who have access to data collections. The algorithm uses input parameters comprising a set of PCDs, a set of agents and their roles (roles associated to agent can be obtained via the $roles$ function introduced earlier). The function returns a possibly empty set of pairs $\langle D, A_D \rangle$, D being a (reference to a) data collection

of a PCD, and $A_D \subseteq A$ a (possibly empty) set of individual agent identities; these are the agents which have access to the various data collections.

Our PCDs and their operational semantics can be used in policing. We relate permissions and prohibitions for data sharing in a pragmatic fashion. Permissions explicitly indicate who can access the data; if the agent is not permitted, then it will not have access to the data and any attempt to access the data will be recorded as a potential violation. According to this view, one would think that prohibitions would no longer be needed since anything that is not explicitly permitted is forbidden. However, prohibitions can be interpreted as permissions being *revoked* under special circumstances. In this interpretation, prohibitions take precendence over permissions, thus making permissions void under certain circumstances. An example would be a permission to access D and a prohibition to rule out its access at certain times.

A mechanism to police data access factoring in this relation was also presented in [Padget and Vasconcelos 2015]. It takes as input a set of PCDs, an agent id a, the set R of roles, an action $\pi^{\mathbf{Act}}$, a target data collection D and a history $\mathcal{H}$. The history is used as a "sliding window" from a state in the past to the current state. The mechanism initially assumes access is prevented, then it carries out an analysis of existing PCDs: it checks if, in the set of PCDs, there is a permission on action $\pi^{\mathbf{Act}}$ concerned with data D (given as a parameter) and with associated role r; it also checks if the permission is currently valid within a window. The mechanism then checks if the permission is applicable to agent a (via one of its roles r_a); if it is, then access is granted (provisionally). We then check if a prohibition on action $\pi^{\mathbf{Act}}$ over D and with associated role r' exists in the set of PCDs; it also checks that the policy is active within the relevant window. If such a PCD exists, then we check if the prohibition applies to a (via one of its roles r_a); if it is applicable, then access is denied, and we record a's attempt to perform $\pi^{\mathbf{Act}}$ in D.

Alternatively, we can regard permissions as exceptions to prohibitions and obligations, that is, they are *strong permissions* [Boella and van der Torre 2003]. We extend our previous definitions of policy compliance to cater for this. A prohibition $\Pi = \langle \forall \vec{x}.\varphi^a, \exists \vec{y}.\varphi^d, \forall \vec{z}.\mathsf{F}\pi \rangle$ was complied in $\mathcal{H} = \langle \mathcal{S}_1, \ldots, \mathcal{S}_n \rangle$, under σ^a and σ^d, if, and only if, these hold: (1) $active(\Pi, \mathcal{H}, \sigma^a, \sigma^d)$, and (2) $\mathbf{I}(\mathcal{S}_j, \exists \vec{z}.(\pi \cdot \sigma^a), \Sigma)$ does not hold for any state $\mathcal{S}_j, 1 < j \leq n$. If (2) is not met, that is, there is a ground action $\bar{\pi} \in \mathcal{S}_j, 1 < j \leq n$, $\bar{\pi} = (\pi \cdot \sigma^a) \cdot \sigma$ for some σ (it unifies with the target of the prohibition), we check if there is an active permission allowing this: if there is a $\Pi_k = \langle \forall \vec{x}_k.\varphi^a_k, \exists \vec{y}_k.\varphi^d_k, \exists \vec{z}_k.\mathsf{P}\pi \rangle$, $active(\Pi_k, \mathcal{H}', \sigma^a_k, \sigma^d_k)$ $\mathcal{H} = \mathcal{H}_1 \bullet \mathcal{H}' \bullet \mathcal{H}_2$, $\mathcal{H}' = \mathcal{H}'_1 \bullet \langle \mathcal{S}_j \rangle \bullet \mathcal{H}'_2$, $\bar{\pi} = (\pi \cdot \sigma^a_k) \cdot \sigma'$ ($\pi \cdot \sigma^a_k$ is unifiable with $\bar{\pi}$), then there was no violation. For those cases when an obligation and a permission overlap (their activation periods and targets), then if the obligation is deactivated while the permission was still active and the target action was not performed, then there is no violation (the permission makes the obligation optional).

4. A COMPUTATIONAL MODEL

The computational counterpart of the formal model set out in Section 3 and specifically the operational semantics in Section 3.4 is realised using the Institutional Action Language (InstAL) [Padget et al. 2016; Cliffe et al. 2005], which in turn is implemented in Answer Set Prolog (AnsProlog). The justification is twofold: (i) InstAL is a domain-specific language for building institutional models, such as, in this case, the regulations governing access to some data, and (ii) InstAL has an underpinning mathematical model and a formal specification [Cliffe 2007] that connects the formal model to the translation of language fragments into AnsProlog, thus providing a sound formal foundation for the policy model.

We summarize the main features of InstAL here to make this article self-contained, but for an extended discussion, see [Padget et al. 2016]. InstAL is inspired by the social institutions described by [North 1990] and the institutional action arena set out in [Ostrom 2005]. Secondly, it draws on two key notions from the literature, namely "counts-as" [John R. Searle 1995], which leads to the distinction between external and institutional events, and institutional power [Jones and Sergot 1996], which determines whether an institutional event affects the institutional state or not, that is, does it really happen, depending on whether the actor has not just the permission but also the power to bring it about[5]. Thirdly, it builds on Action Languages [Gelfond and Lifschitz 1998], the event calculus [Kowalski and Sergot 1986] and the situation calculus [Pinto and Reiter 1995], which establish the idea of fluents – being facts that are true if present and false if not (i.e. closed-world assumption) – where inertial fluents persist from initiation to termination (addressing the frame problem), while non-inertial fluents only hold as long as the condition on which they depend is true.

Thus, InstAL has external and institutional events, and (institutional) states comprising (i) inertial fluents representing domain, permission[6], power and obligation facts, and (ii) non-inertial fluents representing conditions over facts in given state. Hence, by expressing the definitions of the elements of policy-carrying data language in terms of InstAL, we obtain the benefits both of its formal and computational model. Taking each of the PCD language elements in turn:

(1) A state $\mathcal{S}$ corresponds to a list of inertial and non-inertial facts as identified above, such as $\texttt{individual}(\texttt{i285})$, $\texttt{user}(\texttt{u455})$
(2) An event $\bar{\pi}^{\mathbf{Act}}$ corresponds to an (external) InstAL event, such as $\texttt{access}(\texttt{Agent}, \texttt{Dataset})$[7], which depending on the extant permissions and any other conditions over the policy state at the time, may lead to the occurrence of the corresponding (institutional) event, such as $\texttt{intAccess}(\texttt{Agent}, \texttt{Dataset})$, or if the event is not permitted to the violation event $\texttt{viol}(\texttt{access}(\texttt{Agent}, \texttt{Dataset}))$.
(3) A history $\mathcal{H}$ corresponds to a set of (institutional) states, typically labelled by an instant – usually an integer – that denotes the time at which an event was observed and at which time a collection of (institutional) facts hold. Instants simply provide an ordering and are not necessarily connected to a precise notion of the passage of time. The history is the computational consequence of an event trace (operations on the resource as interpreted in terms of the governing policy), as determined by the rules that initiate and terminate fluents or establish the presence or absence of non-inertial fluents.
(4) A policy (Def. 3.4): corresponds to an institution definition in InstAL, which comprises type declarations, event and fluent declarations, generation rules (that determine whether external events count-as institutional events), initiation and termination rules (that determine the consequences for the policy state) and non-inertial rules (that capture dynamic conditions over the policy state). We use the three examples of Fig 2 from Section 3.6 to illustrate how PCDs can be captured in the InstAL framework:
    (a) A permission (definition 3.7): corresponds to InstAL's institutional permission fact, written $\texttt{perm}(\texttt{action})$. In the case of PCD(1) in Fig.2, this is expressed as:

```
initially perm(access(A,D1)), perm(intAccess(A,D1));
```

[5]Just as the chair of a meeting is only one who can the start and end of business.
[6]InstAL offers by default a model in which all actions (events) are prohibited unless explicitly permitted, although the converse is easily defined as demonstrated in [King et al. 2015]
[7]We follow the convention in logic programming that a literal starts with a lower case letter, while a variable starts with a capital.

```
intAccess(A,D1) terminates perm(intAccess(B,D1)) ...
```

where the first line expresses the permission for any agent A to access all of the records in resource D_1, because the permission is universally quantified through the variable in the first position, where access is the exogenous event and intAccess is the corresponding institutional event. The second line indicates that the occurrence of the intAccess event terminates permission to access all the records in resource D_1 for every agent.

(b) A prohibition (definition 3.6): corresponds to the absence of permission to do something in the default InstAL behaviour. Thus, for the example in PCD (2) in Fig. 2, we might assume that initially all agents have permission to read from D_1 and from D_2, but if an agent access the first resource, it may not access the second so that for example the situation described regarding the resource D_1 and D_2 can be captured as:

```
initially perm(access(A,D1)), perm(intAccess(A,D1));
initially perm(access(A,D2)), perm(intAccess(A,D2));
intAccess(A,D1) terminates perm(intAccess(A,D2)) ...
```

which is very similar to the previous example in terms of the initial permissions, but the revocation of permission applies to A in respect of D_2.

(c) An obligation (definition 3.5): the counterpart in InstAL takes the form of an obligation fluent which in its full form is a triple associating a compliance action with a deadline event and a violation event, to indicate that the action must occur before the deadline or a violation occurs. InstAL also allows the specification of a compliant state, whose achievement satisfies the obligation, or a "deadline" state that triggers the violation event. In this fragment, we use a shorthand form of obligation in which we only specify the action that discharges the obligation, since there is no deadline:

```
intAccess(A,D1) initiates obl(provide(A,D2)) ...
```

The purpose of the above is to provide an intuition for the representation of the formal language presented in Section 3, through a mapping of some examples to fragments of InstAL. We now explain the ways in which we use Answer Set Programming, continuing with the examples described in Section 3.6.

4.1. Policies and Answer Set Programming

Before deployment, a policy author would like to know whether the policy does what it is intended to do – in effect, whether it satisifies its requirements. This is a kind of testing, in which (for policies informally described on paper) walk-throughs with use-cases determine whether desired outcomes are achieved and undesired ones avoided. A policy specification in InstAL supports the policy author in two ways: by enabling policy validation off-line (using single-shot solving) and to monitor compliance on-line (using incremental solving)[8].

One form of validation takes specific use cases (presented as traces) that capture desired outcomes, namely the correct handling of policy-compliant behaviour and the detection of non-compliant behaviour (see examples in Section 4.2). This approach however does only validate policy for situations that the policy-maker can anticipate. This may work for simple policies in isolation, where all the possibilities are clear, but loop-

[8]We use the Potsdam Answer Set Solving Collection (Potassco), specifically clingo, available from http://potassco.sourceforge.net/, accessed 2016-09-16.

```
institution example;

type Agent;
type Dataset;
type PCD;
type Role;

fluent role(Agent,Role);
fluent pcd(Dataset,PCD,Role);
fluent accessed(Agent,Dataset,PCD,Role);

exogenous event access(Agent,Dataset);
inst event intAccess(Agent,Dataset);

access(A,D) generates intAccess(A,D);

intAccess(A,D) initiates accessed(A,D,P,R)
   if role(A,R), pcd(D,P,R);
intAccess(A,D) terminates
   perm(access(B,D)), pow(intAccess(B,D)), perm(intAccess(B,D))
   if role(A,R), pcd(D,P,R);
intAccess(A,d1) terminates
   perm(access(A,d2)), pow(intAccess(A,d2)), perm(intAccess(A,d2))
   if role(A,R), pcd(D,P,R);

fluent provided(Agent,Dataset,PCD,Role);
exogenous event provide(Agent,Dataset);
inst event intProvide(Agent,Dataset);

exogenous event forever;
violation event never;
obligation fluent obl(provide(Agent,Dataset),forever,never);

intAccess(A,d1) initiates
   obl(provide(A,d1),forever,never),
   perm(provide(A,d1)),  perm(intProvide(A,d1)),  pow(intProvide(A,d1))
   if role(A,R), pcd(D,P,R);

provide(A,D) generates intProvide(A,D);

intProvide(A,D) initiates provided(A,D,P,R)
   if role(A,R), pcd(D,P,R);
```

Fig. 3. The example policy specification

holes and unintended consequences can all too easily arise as the policy becomes more complicated or interacts with other policies (more on this in Section 6).

A second form of validation helps the policy author address this problem: instead of presenting particular traces, the solver can compute all possible traces of a given length (i.e. a number of instants), for the events defined in the model. Without any constraints, that is all the permutation sequences of length n, many of which may make no sense in the light of domain knowledge, such as whether an event can occur more than once and whether one event can only occur after another (see, for example, [Pieters et al. 2015]). Consequently, the author can specify constraints that capture such domain knowledge and reduce the search space, while also specifying conditions in order to be presented with, say, all traces that lead to good or bad states.

A third form of validation is compliance monitoring, where the same model as above is presented with one event at a time and the solver computes the next state of the model (hence multi-shot or incremental solving). Consequently, violations can be detected and appropriate actions taken when revising the PCD specification.

As we noted in the previous section, a policy Π is expressed as an institutional model using InstAL, which we then instantiate to create a PCD $\langle \Pi, D \rangle$, where Π is grounded with respect to the dataset D and the policy provides the actions `access` and `provide`,

through which an individual operates on the dataset. A sequence of actions and the states they establish are captured as answer sets – using either single or multi-shot solving – which in turn encapsulate each PCD history $\mathcal{H}$. In the next section, we use single-shot solving to explore the behaviour of some illustrative policies, as described in Section 3.6, against some sample traces.

4.2. Policy validation by use case

To demonstrate the computational model of the formalisation presented in section 3, the three example PCDs from Fig. 2 are combined in a single specification (Fig. 3) and usage scenario where Fig. 4 gives an event-oriented view of the events that occur and at which instants given fluents hold, while Fig. 5 gives a state-oriented view, showing which fluents are initiated, hold and are terminated in each state. Note that for an event that occurs at time i, any fluents that it initiates show as holding from time $i + 1$ onward. The `@` notation shows the name of the institution (in this case `example`) that recognises the event and in which the fluents hold. Here there is only one institution, but the visualization tools account for models with multiple institutions.

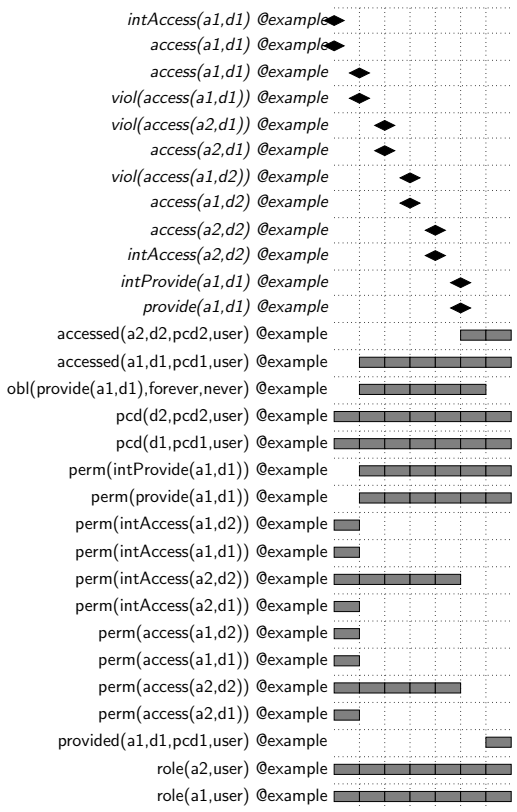

Fig. 4. The example policy event occurrence (denoted by diamonds) and fluent duration chart (grey blocks), time steps run left to right.

As described earlier, PCD(1) captures the permission to access all records of a data collection. The activation condition specifies that the permission is in place if the records have not yet been accessed, and the policy is deactivated when the records are accessed (lines 19–21, Fig. 3); the policy stipulates a "one-off" access to the data, so whereas `A` is bound to the accessing agent, `B` is universally quantified with respect to all agents. All agents associated with the role of `user` may take advantage of this policy. As the trace shows (Fig. 5), the first access to `d1` by `a1` succeeds – logged by the presence of the fluent $\texttt{accessed}(\texttt{a1}, \texttt{d1})$ in the policy state history – but subsequent attempts (both by a_1 and a_2) result in a violation, because all the permissions have been struck out after the first read (see state $\mathcal{S}_0$, where the struck through fluents identify those that are not present in the next state (because they are terminated in this one).

PCD(2) illustrates how to inter-relate policies (see lines 22–24, Fig.3). It states that anyone who accesses `d1` is forbidden to access `d2`. The policy will never be deactivated once it is activated. Thus, once `a1` makes use of the permission established by PCD(1) to access `d1`, its permission to access `d2` is revoked (along with the permissions for any access `d1`, as per PCD(1)), but `a2` can still access `d2` as seen in $\mathcal{S}_5$ of Fig.5.

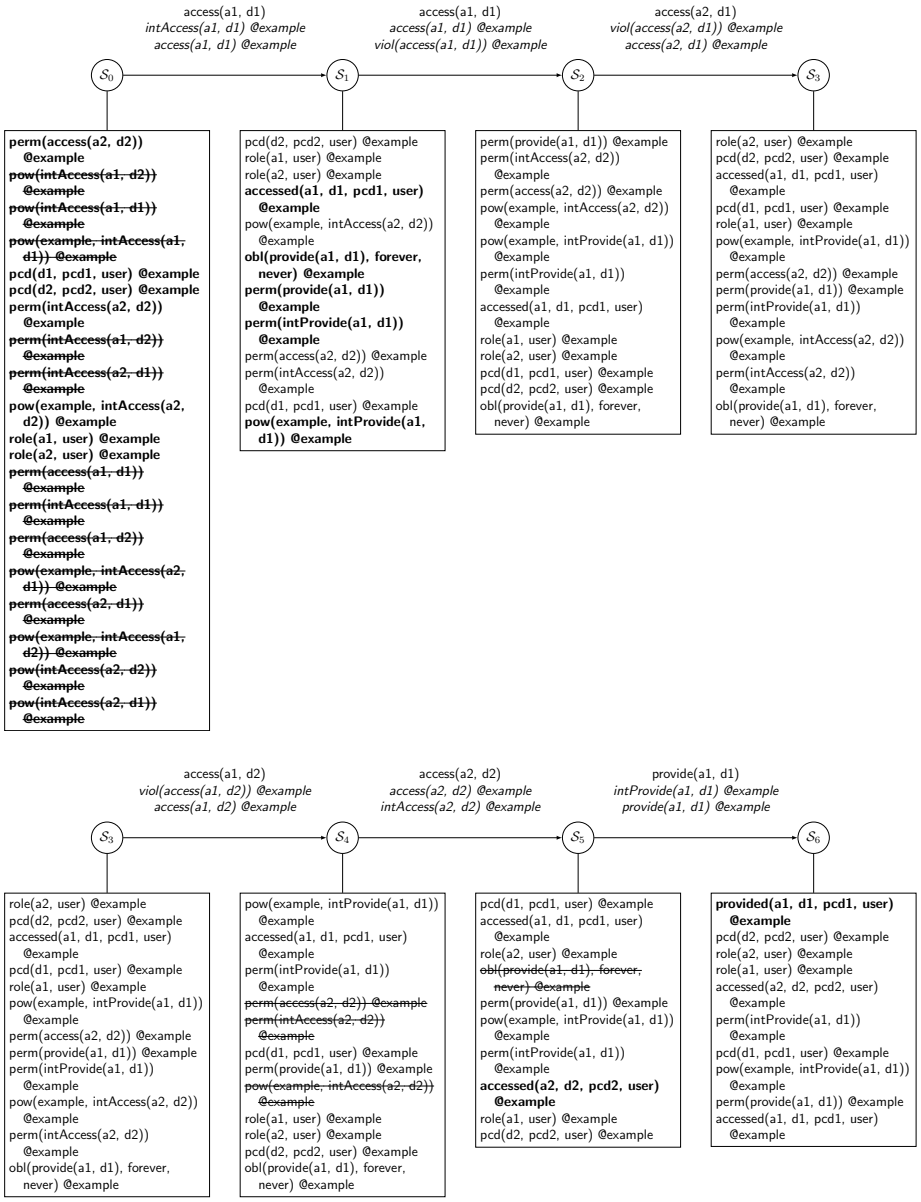

Fig. 5. The example policy trace: states 0–6

Finally, PCD(3) states that an agent that accesses `d1` is obliged to provide records (to be added) to `d2`. The obligation is initiated in state $\mathcal{S}_1$ and is deactivated after the agent who accessed `d1` provides some data in state $\mathcal{S}_6$ (thus it is struck out in $\mathcal{S}_5$ to highlight that it is not present in $\mathcal{S}_6$).

We illustrate above how to validate a policy against given traces that lead to known outcomes. The same model can also be used to validate normative properties of a policy by checking for the (non-)existence of traces that lead to (un)desirable outcomes, by expressing as conditions over events and states, as described in [Hopton et al. 2009].

The details for the scenario set out above are omitted here for lack of space, but the application of the principle can be seen in [Pieters et al. 2015].

5. RELATED WORK

At the macro-level, we are inspired by Berners-Lee's [Berners-Lee 1999] vision of the semantic web as a collection of connected resources that, remarkably for a text about future developments in computing, remains relevant nearly two decades later. More recently, Berners-Lee has called for a bill of rights or magna carta [Kiss 2014] to address issues of privacy, censorship and control of the internet. That is an on-going and evolving debate in the febrile political environment of early 2017, stimulated by the cases of Manning, Snowden and the Democratic National Committee (in the USA), amongst others and the Draft Communications Data Bill (in the UK) which was eventually enacted as the Investigatory Powers Bill [Investigatory Powers Bill 2016]. The proposal here seeks to provide a formalism, associated mechanisms and a computational framework to capture specific features that reflect the principles capturing the notions of privacy preferences and policies as described in [Berners-Lee 1999], but taking into account the broader context that is being created by IoT and STS in the years since.

Research on security and privacy explored alternatives for authentication and authorisation, including the popular role-based access control (RBAC) models [Sandhu et al. 1996; Suhendra 2011], building on role theory [Biddle 1979; Turner 2001]. These assume, however, that the principal can only act on the subject in a context where the principal's actions can be observed and controlled. This clearly does not hold in an environment in which data is shared and propagated largely without oversight, although [Cheng et al. 2012; Karjoth et al. 2003] begin to address this scenario. Nevertheless, once the data is outside the domain in which the policy can be enforced, the guarantees that a security framework such as RBAC provides almost certainly cannot be upheld and encryption probably only delays access. Thus, expectations about the treatment of data must be revised to accept transparency in place of privacy, although this too cannot necessarily be assured. Some of the practicalities arising from this are discussed in [Sackmann and Kähmer 2008]. Hansen [2012] sets out higher level requirements: "unlinkability when possible and desired, transparency on possible and actual linkages, and the feasibility for data subjects to exercise control or at least intervene in the processing of data." We notice "where possible": there cannot be absolute guarantees, only best efforts.

Others have independently used the term "policy-carrying data". The research presented in [Wang et al. 2013] and [Saroiu et al. 2015] introduces concepts by the same name, but their focus is on encryption aspects, architecture and information models, and how their approaches can be implemented/integrated with specific technologies. There is very little detail about the policy languages they might support and no discussion of their semantics, formalisation or the scope for reasoning about policy as a normative framework, as described here. As mentioned before, we build upon, expand and adapt our work presented in [Padget and Vasconcelos 2015], in which a much simpler formalisation in propositional logic was presented. Our present research offers a first-order logic formalism, with practical concerns – our language is not as expressive as full first-order logic, but the associated mechanisms are decidable.

Our work draws upon research on normative multi-agent systems [Andrighetto et al. 2013], especially on proposals for norm specification [Savarimuthu et al. 2013; Şensoy et al. 2012; Vasconcelos et al. 2009] and normative (practical) reasoning [Balke et al. 2013; García-Camino et al. 2009; Meneguzzi et al. 2015]. Our notation is heavily inspired by existing work [García-Camino et al. 2009; Şensoy et al. 2012; Vasconcelos et al. 2009] but we simplify the components of our policies, leaving out aspects such as

deadlines and sanctions/rewards. A rule-based language such as [García-Camino et al. 2009], being Turing-complete, would allows us to represent arbitrary concepts, but its expressiveness would render reasoning mechanisms more complex. We note that our semantics – the explicit recording of states of the computation – has been used in the literature, either as Kripke structures (providing the usual underpinning of modal deontic logics [McNamara 2006]) or as operational semantics [García-Camino et al. 2009; Vasconcelos et al. 2012].

We also report on [Karjoth et al. 2003], which describes a platform to enforce individual enterprise privacy "promises" across multiple enterprises. The work presents useful practical examples of obligations, such as "we delete collected data if consent is not given within 15 days", and 4 stakeholders/roles are identified, namely (i) data subject, (ii) data users, (iii) privacy officer, and (iv) security officer. A mapping translates application-independent obligations into available actions, so there is an abstract (institution-like) layer, but this is not recognised explicitly as a concept. A useful contribution is the notion of "sticky policies" associated with data in the same way as metadata. Their formalisation adopts the Authorization Specification Language of [Jajodia et al. 2001].

6. CONCLUSIONS, DISCUSSION AND FUTURE WORK

This paper draws upon the extensive body of research on normative (multi-agent) systems to propose a formal framework based on Deontic first-order logic to represent and reason with/about data access policies. The application of principles from normative systems gives rise to a language that can be seen as "sufficiently rich" – in that it is known to be adequate to capture norms – as well as one that is "agent-oriented", making the approach suitable for complex socio-technical systems.

The main idea is that a policy conceptually encapsulates a data resource, to give the notion of policy-carrying data (PCD). This does not imply physical encapsulation, since that would then preclude making a single resource subject to multiple policies (e.g. depending on the role of the accessor or other factors). Furthermore, we assume and do not address data en/decryption, but observe that the combination of policy and (encrypted) data offers a kind of quasi-homomorphic encryption (the policy enforces the operations that can be carried out), in contrast to full homomorphic encryption (the form of the encryption is what ensures only permitted operations work).

Some elements of future work are quite straightforward and follow from recent work on connected and interacting institutions [Padget et al. 2016], and on hierarchical institutions [King et al. 2015]. The examples presented in sections 3.6 and 4.2 show a policy that associates access to one dataset with access to another. This illustrates how a single policy might be used to control access to two resources. In contrast, an important aspect to address in future work is policy interaction, where actions taken in the context of one policy have an effect in one or more others, such as expanding or limiting an actor's range of permitted actions or incurring obligations. Provision for policy interaction is a practical necessity, because one policy for everything has no sense and because it is both desirable and inevitable that policies will be developed and revised independently and incrementally.

We must also draw attention to some limitations in our proposal. In particular, we acknowledge there are situations for which our formalism is not adequate or simply not expressive enough. For instance, for situations in which policies are addressed to *groups* of users, as studied in, *e.g.* [Aldewereld et al. 2016], our formalism and its (operational) semantics may be awkward. More concretely, if we need to represent, say, an obligation on m individuals to provide as a group n records (that is, the obligation is fulfilled if one or more individuals in the group provide n records, and not $m \times n$ records), we will need to create m obligations – one for each individual – and their de-

activation conditions would be if anyone (possibly more than one individual) provides n records. We also note that more subtle normative aspects, *e.g.*, differentiating permissions and rights – where a right (of someone) implies in an obligation for someone else – would require "chains" of policies whose violation/compliance may prove hard to detect.

We would like to extend our formalism to represent rewards/sanctions when policies are complied with or violated. These rewards/sanctions add a game-theoretic aspect through utilities, which should be factored in when stakeholders design and reason with/about policies. This is currently being investigated within a peer-to-peer scenario [Cauvin et al. 2016]. Additionally, we are aware that active policies can be useful when establishing the context (activation and deactivation conditions) of other policies, as explored in, for instance [García-Camino et al. 2009]. We will explore means to extend our formalism to enable us to represent active policies as part of the activation and deactivation conditions.

An important issue that we have not addressed in this presentation is the technical means to ensure that actions and events are reliably logged for auditing or use in post-mortem analyses. Basin et al. [2013] among others point out various problems with incomplete and disagreeing logs and provide means to handle these in a centralised fashion. Although we introduced our approach using a centralised model in Section 2 and assumed we have access to complete (ever-growing) histories of events, these are not realistic, failing to scale up and creating bottlenecks and single-points of failure. Alternative distributed approaches such as the one reported in, for instance, [Vasconcelos et al. 2012], could be adapted/extended for our purposes. Additionally, properties of various experimental forms of distributed ledger – those focussing on data and contracts, rather than value transfer like Blockchain [Underwood 2016] – appear promising and are being investigated [Cauvin et al. 2016], so that participants hold encrypted copies of relevant events and histories thus bypassing central servers/repositories.

Finally, although the computational representation of our policies are declarative, authoring such specifications requires experience and specialist knowledge. We are therefore exploring the possibility of using controlled natural language to write regulations, inspired by narrative theory [Thompson et al. 2015].

REFERENCES

Huib Aldewereld, Virginia Dignum, and Wamberto W. Vasconcelos. 2016. Group Norms for Multi-Agent Organisations. *ACM Trans. Auton. Adapt. Syst.* 11, 2 (2016).

Ross J. Anderson. 2001. *Security Engineering: A Guide to Building Dependable Distributed Systems* (1st ed.). John Wiley & Sons, New York, NY, USA.

Giulia Andrighetto, Guido Governatori, Pablo Noriega, and Leendert W. N. van der Torre (Eds.). 2013. *Normative Multi-Agent Systems*. Dagstuhl Follow-Ups, Vol. 4. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany. i–xi pages.

Krzysztof R. Apt. 1997. *From logic programming to Prolog*. Prentice Hall, London.

Tina Balke, Marina De Vos, and Julian A. Padget. 2013. Evaluating the Cost of Enforcement by Agent-Based Simulation: A Wireless Mobile Grid Example, See [Boella et al. 2013]. DOI:http://dx.doi.org/10.1007/978-3-642-44927-7_3

David Basin, Felix Klaedtke, Srdjan Marinovic, and Eugen Zălinescu. 2013. Monitoring Compliance Policies over Incomplete and Disagreeing Logs. In *Runtime Verification*. LNCS, Vol. 7687. Springer.

David Basin, Felix Klaedtke, and Samuel Müller. 2010. Policy Monitoring in First-Order Temporal Logic. In *Procs. 22nd Int'l Conf. on Computer Aided Verification (CAV 2010) (LNCS)*, Vol. 6141. Springer Verlag, 1–18.

Tim Berners-Lee. 1999. *Weaving the Web: The Past, Present and Future of the World*

Wide Web by its Inventor. Orion Business. ISBN-13: 978-0752820903.
Bruce J. Biddle. 1979. *Role Theory*. Academic Press, San Diego.
Guido Boella, Edith Elkind, Bastin Tony Roy Savarimuthu, Frank Dignum, and Martin K. Purvis (Eds.). 2013. *Procs. Principles & Practice of Multi-Agent Systems (PRIMA)*. LNCS, Vol. 8291. Springer.
Guido Boella and Leendert van der Torre. 2003. Permissions and Obligations in Hierarchical Normative Systems. In *Procs. 9th Int'l Conf. on A.I. & Law (ICAIL '03)*. ACM, 109–118. DOI:http://dx.doi.org/10.1145/1047788.1047818
Laura Brandimarte, Alessandro Acquisti, and George Loewenstein. 2013. Misplaced Confidences: Privacy and the Control Paradox. *Social Psychological and Personality Science* 4, 3 (2013), 340–347.
Bruce G. Buchanan and Richard O. Duda. 1983. Principles of Rule-Based Expert Systems. In *Advances In Computers*, Marshall C. Yovits (Ed.). Advances in Computers, Vol. 22. Elsevier, 163 – 216. DOI:http://dx.doi.org/10.1016/S0065-2458(08)60129-1
Claudio Castellini. 2005. *Automated Reasoning in Quantified Modal and Temporal Logics*. Ph.D. Dissertation. School of Informatics, University of Edinburgh.
Samuel R. Cauvin, Martin J. Kollingbaum, Derek Sleeman, and Wamberto W. Vasconcelos. 2016. Towards a Distributed Data-Sharing Economy. Int'l Workshop on Coordination, Organizations, Institutions and Norms (COIN@ECAI-2016). (2016).
Yuan Cheng, Jaehong Park, and Ravi S. Sandhu. 2012. A User-to-User Relationship-Based Access Control Model for Online Social Networks. In *Data and Applications Security and Privacy XXVI - 26th Annual IFIP WG 11.3 Conference, DBSec 2012 (LNCS)*, Nora Cuppens-Boulahia, Frédéric Cuppens, and Joaquín García-Alfaro (Eds.), Vol. 7371. Springer, 8–24. DOI:http://dx.doi.org/10.1007/978-3-642-31540-4_2
Owen Cliffe. 2007. *Specifying and analysing institutions in multi-agent systems using answer set programming*. Ph.D. Dissertation. University of Bath. http://opus.bath.ac.uk/16762/
Owen Cliffe, Marina De Vos, and Julian A. Padget. 2005. Specifying and Analysing Agent-Based Social Institutions Using Answer Set Programming. In *Agents, Norms and Institutions for Regulated Multi-Agent Systems, ANIREM 2005, and Organizations in Multi-Agent Systems, OOOP 2005, Revised Selected Papers (LNCS)*, Olivier Boissier, Julian A. Padget, Virginia Dignum, Gabriela Lindemann, Eric T. Matson, Sascha Ossowski, Jaime Simão Sichman, and Javier Vázquez-Salceda (Eds.), Vol. 3913. Springer, 99–113. DOI:http://dx.doi.org/10.1007/11775331_7
Murat Şensoy, Timothy J. Norman, Wamberto W. Vasconcelos, and Katia Sycara. 2012. OWL-POLAR: A framework for semantic policy representation and reasoning. *Web Semantics: Science, Services and Agents on the World Wide Web* 12-13 (2012).
David Ferraiolo, Vijayalakshmi Atluri, and Serban Gavrila. 2011. The Policy Machine: A novel architecture and framework for access control policy specification and enforcement. *Journal of Systems Architecture* 57, 4 (2011).
Michael Fisher. 2006. METATEM: The Story So Far. In *Procs. of 3rd Int'l Conf. on Programming Multi-Agent Systems (ProMAS'05) (LNAI)*, Vol. 3862. Springer-Verlag, 3–22. DOI:http://dx.doi.org/10.1007/11678823_1
Melvin Fitting. 1996. *First-order Logic and Automated Theorem Proving* (2 ed.). Springer-Verlag New York, Inc., Secaucus, NJ, USA.
Andrés García-Camino, Juan A. Rodríguez-Aguilar, Carles Sierra, and Wamberto W. Vasconcelos. 2009. Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems* 18, 1 (2009), 186–217.
Michael Gelfond and Vladimir Lifschitz. 1998. Action Languages. *Electron. Trans. Artif. Intell.* 2 (1998), 193–210.
Guido Governatori, Francesco Olivieri, Antonino Rotolo, and Simone Scannapieco. 2013. Computing Strong and Weak Permissions in Defeasible Logic. *Journal of*

Philosophical Logic 42, 6 (2013), 799–829. http://www.jstor.org/stable/42001261
Marit Hansen. 2012. Top 10 Mistakes in System Design from a Privacy Perspective and Privacy Protection Goals. In *Privacy and Identity Management for Life*. IFIP Adv. in Inf. & Comm. Techn., Vol. 375. Springer, 14–31.
Luke Hopton, Owen Cliffe, Marina De Vos, and Julian A. Padget. 2009. AQL: A Query Language for Action Domains Modelled Using Answer Set Programming *(LNCS)*, Esra Erdem, Fangzhen Lin, and Torsten Schaub (Eds.), Vol. 5753. Springer. DOI:http://dx.doi.org/10.1007/978-3-642-04238-6_39
Investigatory Powers Bill 2016. UK Legislation. (2016). http://www.legislation.gov.uk/id?title=Investigatory+Powers+Act+2016, retrieved 2017-02-27.
Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. 2001. Flexible support for multiple access control policies. *ACM Trans. Database Syst.* 26, 2 (June 2001), 214–260. DOI:http://dx.doi.org/10.1145/383891.383894
John R. Searle. 1995. *The Construction of Social Reality*. Allen Lane, Penguin Press.
Andrew J. I. Jones and Marek J. Sergot. 1996. A Formal Characterisation of Institutionalised Power. *Logic Journal of the IGPL* 4, 3 (1996), 427–443.
Günter Karjoth, Matthias Schunter, and Michael Waidner. 2003. Platform for enterprise privacy practices: privacy-enabled management of customer data. In *Procs. 2nd Int'l Conf. on Privacy-enhancing technologies (PET'02*. Springer.
Thomas Christopher King, Tingting Li, Marina De Vos, Virginia Dignum, Catholijn M. Jonker, Julian Padget, and M. Birna van Riemsdijk. 2015. A Framework for Institutions Governing Institutions. In *Procs. Int'l Conf. on Autonomous Agents & Multi-agent Systems (AAMAS)*. 473–481. http://dl.acm.org/citation.cfm?id=2772940
Jemima Kiss. 2014. An online Magna Carta: Berners-Lee calls for bill of rights for web. Web content. (March 2014). http://www.theguardian.com/technology/2014/mar/12/online-magna-carta-berners-lee-web, retrieved 20141218.
Robert A. Kowalski and Marek J. Sergot. 1986. A Logic-based Calculus of Events. *New Generation Comput.* 4, 1 (1986), 67–95.
Tingting Li, Tina Balke, Marina De Vos, Julian A. Padget, and Ken Satoh. 2013. A model-based approach to the automatic revision of secondary legislation. In *International Conference on Artificial Intelligence and Law*, Enrico Francesconi and Bart Verheij (Eds.). ACM, 202–206. http://doi.acm.org/10.1145/2514601.2514627
Bernard Litaer. 2002. *The Future of Money: Creating New Wealth, Work and a Wiser World*. Century.
Alberto Martelli and Ugo Montanari. 1982. An Efficient Unification Algorithm. *ACM Trans. Program. Lang. Syst.* 4, 2 (April 1982), 258–282.
Paul McNamara. 2006. Deontic logic. In *Logic and the Modalities in the Twentieth Century*. Vol. 7. North-Holland.
Felipe Meneguzzi, Odinaldo Rodrigues, Nir Oren, Wamberto W. Vasconcelos, and Michael Luck. 2015. BDI reasoning with normative considerations. *Eng. App. of Art. Int.* 43 (2015), 127 – 146.
John-Jules C. Meyer, Frank P. M. Dignum, and Roel J. Wieringa. 1994. *The paradoxes of deontic logic revisited: a computer science perspective*. Technical Report UU-CS-1994-38. University of Utrecht, Utrecht.
John-Jules C. Meyer and Roel. J. Wieringa. 1993. Applications of Deontic Logic in Computer Science: A Concise Overview. In *Deontic Logic in Computer Science: Normative System Specification*. John Wiley & Sons.
Douglass C North. 1990. *Institutions, institutional change and economic performance*. Cambridge university press.
Elinor Ostrom. 2005. *Understanding Institutional Diversity*. Princeton Universiy Press. ISBN: 9780691122380.

P3P 2006. The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. World Wide Web Consortium (W3C). (November 2006). https://www.w3.org/TR/P3P11/ Retrieved 2017-02-27.

Julian Padget, Emad ElDeen Elakehal, Tingting Li, and Marina De Vos. 2016. *InstAL: An Institutional Action Language*. Springer International Publishing, 101–124. DOI:http://dx.doi.org/10.1007/978-3-319-33570-4_6

Julian Padget and Wamberto W. Vasconcelos. 2015. Policy-Carrying Data: A Step Towards Transparent Data Sharing. *Procedia Computer Science* 52 (2015), 59 – 66.

Wolter Pieters, Julian Padget, Francien Dechesne, Virginia Dignum, and Huib Aldewereld. 2015. Effectiveness of qualitative and quantitative security obligations. *Journal of Information Security and Applications* 22 (2015), 3–16. DOI:http://dx.doi.org/10.1016/j.jisa.2014.07.003

Javier Pinto and Raymond Reiter. 1995. Reasoning About Time in the Situation Calculus. *Ann. Math. Artif. Intell.* 14, 2-4 (1995), 251–268.

Raymond Reiter. 1978. On Closed World Databases. In *Logic and Databases*. Plenum Press, NY, USA.

Stefan Sackmann and Martin Kähmer. 2008. ExPDT: A Policy-based Approach for Automating Compliance. *Wirtschaftsinformatik/Angewandte Informatik* 50 (2008), 366–374. Issue 5.

Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. 1996. Role-Based Access Control Models. *Computer* 29, 2 (Feb. 1996), 38–47.

Stefan Saroiu, Alec Wolman, and Sharad Agarwal. 2015. Policy-Carrying Data: A Privacy Abstraction for Attaching Terms of Service to Mobile Data. In *HotMobile'15*. ACM Press.

Bastin Tony Roy Savarimuthu, Julian Padget, and Maryam Purvis. 2013. Social Norm Recommendation for Virtual Agent Societies, See [Boella et al. 2013], 308–323.

Vivy Suhendra. 2011. A Survey on Access Control Deployment. In *Security Technol.* Comm. in Comp. & Inf. Science, Vol. 259. Springer.

Matthew Thompson, Julian Padget, and Steve Battle. 2015. Governing Narrative Events With Institutional Norms. In *6th Workshop on Computational Models of Narrative, CMN 2015 (OASICS)*, Mark A. Finlayson, Ben Miller, Antonio Lieto, and Rémi Ronfard (Eds.), Vol. 45. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 142–151. DOI:http://dx.doi.org/10.4230/OASIcs.CMN.2015.142

Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok. 2003. Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In *Procs. ISWC 2003*. LNCS, Vol. 2870. Springer.

Ralph H. Turner. 2001. *Role Theory*. Springer US, Boston, MA, 233–254.

Sarah Underwood. 2016. Blockchain Beyond Bitcoin. *Commun. ACM* 59, 11 (Oct. 2016), 15–17. DOI:http://dx.doi.org/10.1145/2994581

Wamberto Weber Vasconcelos, Andrés García-Camino, Dorian Gaertner, Juan A. Rodríguez-Aguilar, and Pablo Noriega. 2012. Distributed norm management for multi-agent systems. *Expert Syst. & Appl.* 39, 5 (2012), 5990–5999.

Wamberto W. Vasconcelos, Martin J. Kollingbaum, and Timothy J. Norman. 2009. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 19, 2 (2009), 124–152.

Georg H. von Wright. 1951. Deontic Logic. *Mind* 60, 237 (1951).

Xiaoguang Wang, Qi Yong, Yuehua Dai, Jianbao Ren, and Zhang Hang. 2013. Protecting Outsourced Data Privacy with Lifelong Policy Carrying. In *IEEE Int'l Confs. on High Perf. Comp. & Comm. and Embedded & Ubiquitous Comp. (HPCC-EUC)*.